MICRO SOLUTIONS FOR COMPLEX PROBLEMS

μTECH

MANUAL DA QUALIDADE

μ CUBE SURFACE - PT INOVAÇÃO

11/04/2009

AUTORES

André Cardoso
Girson Monteiro
Marcelo Barreira

Histórico de versões

Data	Versão	Descrição	Autor
2009-03-08	0.1	Estruturação do Manual de Qualidade	Girson Monteiro
2009-03-16	0.2	Introdução	Marcelo Barreira
2009-03-16	0.3	Processo de Desenvolvimento Software: -Metodologia -Testes a efectuar	André Cardoso
2009-03-18	0.4	Definição de processos: -Objectivos -Plano do projecto	Girson Monteiro
2009-03-18	0.5	Gestão de riscos: -Politicas de gestão de riscos -Responsabilidades	André Cardoso
2009-03-18	0.6	Apresentação da empresa	Marcelo Barreira
2009-03-19	0.7	Processo desenvolvimento software: -Fases do projecto	André Cardoso
2009-03-19	0.8	Definição de processos: -Plano testes aceitação -Protótipo interface -Relatório arquitectura -Manual de Qualidade	Girson Monteiro
2009-03-21	0.9	Definição de processos: -Relatório de execução da 1ª Fase do Projecto -Relatório de Desenvolvimento -Relatório de Testes	Girson Monteiro
2009-03-22	1.0	Normas, métricas e análise	André Cardoso
2009-03-24	1.1	Definição de processos: -Manual do utilizador -Pacotes de Instalação e Manutenção do Produto	Girson Monteiro
2009-03-24	1.2	Resumo, Ferramentas	André Cardoso

2009-03-24	1.3	Glossário, Referências	Girson Monteiro
2009-03-29	1.4	Revisão do documento, Normas de controlo de versões e alterações	Girson Monteiro

RESUMO

O Manual da Qualidade define as principais linhas orientadoras a serem seguidas por todos os elementos da empresa. A adequação das práticas da empresa às metodologias aqui definidas permitirá a conformidade dos produtos produzidos com os padrões de qualidade da empresa. O manual especificará, entre outras, normas de documentação, métricas e processos que devem ser aplicados a todas as actividades da empresa.

ÍNDICE

1. Introdução ---------- 9
1.1. Âmbito e objectivos ---------- 9
1.2. Estrutura ---------- 9
1.3. Autores ---------- 10
2. Apresentação da empresa ---------- 11
2.1. Empresa ---------- 11
2.2. Unidades organizacionais e responsabilidades ---------- 11
3. Processo de desenvolvimento de software ---------- 14
3.1. Metodologia ---------- 14
3.2. Testes a Efectuar ---------- 15
3.2.1. Testes Unitários ---------- 15
3.2.2. Testes de Integração ---------- 16
3.2.3. Testes de Usabilidade ---------- 16
3.2.4. Testes de Aceitação ---------- 16
3.3. Gestão de Riscos ---------- 16
3.3.1. Políticas de Gestão de Riscos ---------- 16
3.3.2. Identificação e avaliação dos riscos ---------- 17
3.3.3. Responsabilidades ---------- 19
3.4. Fases do Projecto ---------- 20
4. Definição de processos ---------- 21
4.1. Objectivos ---------- 21
4.2. Plano do Projecto ---------- 22
4.3. Relatório de especificação de requisitos ---------- 23
4.4. Plano de testes de aceitação ---------- 24
4.5. Protótipo de interface ---------- 24
4.6. Relatório de arquitectura ---------- 25
4.7. Manual de qualidade ---------- 26
4.8. Relatório de execução da 1ª Fase do Projecto ---------- 27
4.9. Manual do utilizador ---------- 28
4.10. Pacotes de Instalação e Manutenção do Produto ---------- 28
4.11. Relatório de Desenvolvimento ---------- 29
4.12. Relatório de Testes ---------- 29
5. Normas, Métricas e Análise ---------- 30
5.1. Objectivos ---------- 30

5.2. Normas de Documentação ---------- 30
5.3. Normas de Codificação ---------- 30
5.4. Normas de controlo de versões e alterações ---------- 33
5.5. Métricas ---------- 33
5.6. *Checklists* de Revisão de Código ---------- 35
6. Ferramentas ---------- 38
6.1. Ferramentas Utilizadas ---------- 38
6.2. Linguagens Utilizadas ---------- 39
6.3. Ferramentas de Desenvolvimento ---------- 39
7. Glossário ---------- 40
8. Referências ---------- 41

Índice de Tabelas

Tabela 1 - Tabela de Tarefas ---- 10
Tabela 2 - *Process Script* Plano do projecto ---- 22
Tabela 3 - *Process Script* Relatório de especificação de requisitos ---- 23
Tabela 4 - *Process Script* Plano testes de aceitação ---- 24
Tabela 5 - *Process Script* Protótipo de interface ---- 24
Tabela 6 - *Process Script* Relatório de arquitectura ---- 25
Tabela 7 - *Process Script* Manual de qualidade ---- 26
Tabela 8 - *Process Script* Relatório de execução da 1ª Fase do Projecto ---- 27
Tabela 9 - *Process Script* Manual do utilizador ---- 28
Tabela 10 - *Process Script* Pacotes de Instalação e Manutenção do Produto ---- 28
Tabela 11 - *Process Script* Relatório de Desenvolvimento ---- 29
Tabela 12 - *Process Script* Relatório de Testes ---- 29
Tabela 13 - Defeitos e Gravidades ---- 35
Tabela 14 - *Checklist* de revisão da estrutura do código ---- 36
Tabela 15 - *Checklist* de revisão da documentação do código ---- 36
Tabela 16 - *Checklist* de revisão das variáveis do código ---- 36
Tabela 17 - *Checklist* de revisão de operações aritméticas do código ---- 37
Tabela 18 - *Checklist* de revisão de ciclos do código ---- 37
Tabela 19 - Checklist de revisão de programação defensiva do código ---- 37

Índice de Figuras

Figura 1 - Estrutura organizacional----------12
Figura 2 - Diagrama de *Extremme Programming*----------14

1. Introdução

Todos os projectos devem seguir certas normas e métricas no processo de desenvolvimento de forma a garantir a qualidade do produto a ser desenvolvido bem como as necessidades e expectativas do cliente. O modo como o desenvolvimento do projecto se define bem como todo o envolvente em torno do mesmo, são fulcrais para o seu sucesso.

Neste manual irão ser descritos todos os processos e metodologias pelos quais a empresa se irá guiar no desenvolvimento da aplicação. Durante o desenvolvimento deste projecto, este manual servirá de orientação aos colaboradores da empresa de forma a assegurar padrões de qualidade a todos os níveis.

1.1. Âmbito e objectivos

A empresa µTech têm como principal objectivo satisfazer o cliente, através da concepção de software de alta qualidade e fiabilidade, bem como crescer de uma forma sustentável e lucrativa. Sendo uma empresa nova, adquirir experiência a nível do mercado do mundo real, também constitui um dos principais objectivos.

Este primeiro projecto aparece através da PT Inovação surgindo no âmbito do desenvolvimento de uma aplicação para incorporar no CubeNavigator, aplicação já desenvolvida pela mesma. Actualmente, quando é necessário reportar dados para análise posterior, isso é feito manualmente por operadores da PT Inovação. Com a nossa aplicação, a construção dos relatórios para análise será feita automaticamente, originando maior liberdade para a análise de dados por parte da PT Inovação. A concepção deste manual servirá de suporte a ser guiado por todos os colaboradores da empresa nos projectos futuros, de forma a assegurar os padrões de qualidade ao longo dos tempos.

1.2. Estrutura

Este manual encontra-se estruturado da seguinte forma:

- A Introdução, onde são referidos os objectivos da empresa e o âmbito do projecto, a estrutura do manual e os autores do mesmo.
- Na segunda secção é apresentada a empresa bem como os cargos dos colaboradores da empresa e responsabilidades respectivas.
- Na terceira secção são especificados os processos utilizados no desenvolvimento da aplicação, bem como a política de testes a que a aplicação irá ser submetida. Nesta mesma secção são analisados os riscos do projecto e especificadas as fases de desenvolvimento do mesmo.
- Na quarta secção é descrita toda a documentação que a empresa irá realizar ao

longo do projecto. O plano do projecto também pode ser consultado nesta secção.

- Na quinta secção são especificadas as normas e as métricas utilizadas na documentação do projecto bem como na codificação do mesmo.
- Na sexta secção são apresentadas as ferramentas e linguagens utilizadas no desenvolvimento de toda a aplicação.
- Na sétima secção é possível consultar o glossário
- A oitava secção é relativa às referências.

1.3. Autores

Os colaboradores da empresa que foram alocados à elaboração do Manual de Qualidade podem ser consultados na tabela abaixo.

Nome	Tarefa desempenhado	Cargo na Empresa
André Cardoso	Estruturação e elaboração do manual	Gestor de Suporte
Girson Monteiro	Estruturação e elaboração do manual	Gestor de Qualidade
Marcelo Barreira	Elaboração do manual	Gestor de Processos

Tabela 1 - Tabela de Tarefas

2. Apresentação da empresa

2.1. Empresa

A empresa surge no âmbito da cadeira de Laboratório de Gestão de Projectos do curso de Mestrado Integrado em Engenheira Informática e Computação (MIEIC) no dia 2 de Março. Esta empresa é constituída por um grupo de estudantes, que se achando capazes, não se resguardaram das dificuldades já evidenciadas pelo mundo de trabalho, e criaram esta empresa de dimensões pequenas. A nível técnico, os elementos desta empresa possuem variadas capacidades na área da Informática adquiridas ao longo do currículo académico. As ideias inovadoras destes jovens colaboradores são uma das principais armas desta empresa, que têm a sua área de negócio bem definida e interiorizada pelos membros. Sem fins lucrativos, esta área de negócio está centrada na satisfação das necessidades dos clientes através da elaboração de aplicações práticas e de fácil usabilidade.

2.2. Unidades organizacionais e responsabilidades

A µTech é constituída por 12 elementos onde cada um dos elementos é responsável por um departamento da empresa. A organização interna da empresa segue a metodologia TSP (*Team Software Process*). Este método é aconselhado a equipas com um elevado número de colaboradores, para projectos de elevado grau de complexidade. Neste método de desenvolvimento de software iterativo, a cada elemento da equipa é atribuído um cargo para que todos desempenhem um papel activo ao longo do desenvolvimento de um projecto. Usando este método, pretende-se assegurar uma maior qualidade e segurança no software desenvolvido bem como melhorar os processos de gestão de desenvolvimento do software dentro da organização. Os elementos foram alocados de forma a tirar o maior proveito dos conhecimentos de cada um nas diversas áreas.

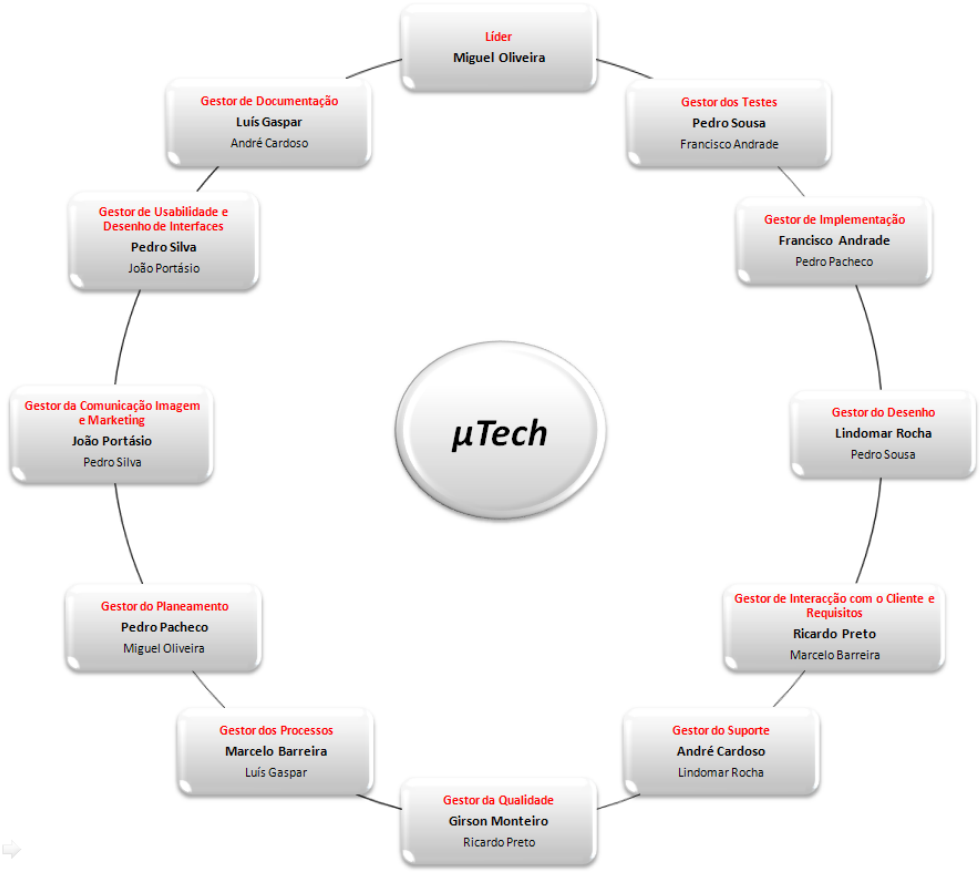

Figura 1 - Estrutura organizacional

Gestor de Testes

Responsável pela realização e documentação dos testes efectuados no software.

Gestor de Implementação

Responsável pela estruturação e implementação do software a ser produzido.

Gestor de Desenho

Responsável pelo desenho da arquitectura dos projectos.

Gestor de Interacção com o Cliente

Responsável pelo levantamento, análise e validação dos requisitos de software.

Gestor de Suporte

Responsável por providenciar todo o tipo de suporte físico necessário ao desenvolvimento de software.

Gestor de Qualidade

Responsável por garantir que todos os padrões idealizados estão a ser seguidos no desenvolvimento de software.

Gestor de Processos

Responsável por organizar e monitorizar todos os processos de desenvolvimento de software.

Gestor de Planeamento

Responsável por estimar as datas previstas de desenvolvimento e finalização de todos os processos durante o desenvolvimento de software.

Gestor de Comunicação, Imagem e Marketing

Responsável por valorizar os recursos intangíveis da empresa para o exterior.

Gestor de Usabilidade e Desenho de Interface

Responsável pelo estudo e desenvolvimento dos desenhos das interfaces do software.

Gestor de Documentação

Responsável por todo o tipo de documentação elaborada ao longo do desenvolvimento de software.

3. Processo de desenvolvimento de software

A fase de desenvolvimento é uma etapa crucial do projecto, e uma forma de garantir o seu sucesso e qualidade é definir um processo de desenvolvimento de software que consiste num conjunto coerente de actividades para especificar, projectar, implementar e testar sistemas de software.

3.1. Metodologia

A µTech adoptou como metodologia de desenvolvimento o *Extremme Programming* (XP). Esta metodologia ágil permite-nos ir rapidamente ao encontro das funcionalidades desejadas pelo cliente, implementando pequenas funcionalidades a cada iteração e obtendo o feedback imediato do cliente acerca do módulo desenvolvido. Cada iteração de desenvolvimento adiciona já uma nova funcionalidade, completamente testada, à versão mais recente do produto. Assim, o desenvolvimento é focado intensamente no *feedback* do cliente, assumindo que os requisitos não são estáticos e apostando em desenvolver a aplicação o mais próximo do cliente possível de modo que cada iteração produza uma nova funcionalidade completamente aceite pelo cliente.

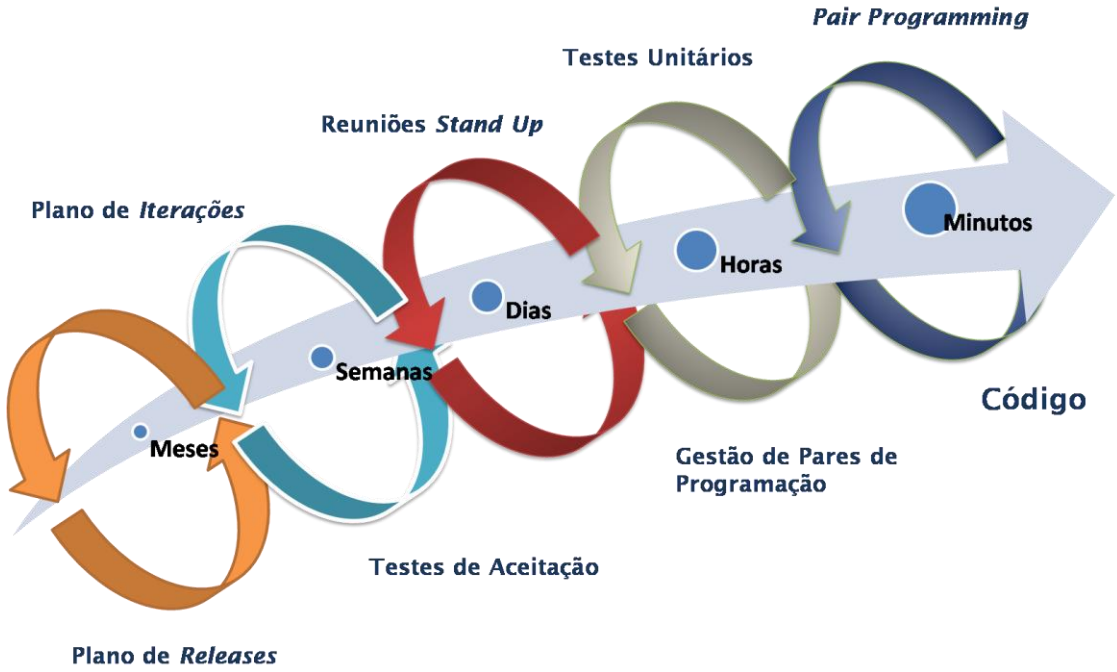

Figura 2 - Diagrama de *Extremme Programming*

As fases de implementação decorrerão em sessões de *Pair Programming*, nas quais 2 elementos da equipa trabalharão juntos numa única tarefa, correspondente a uma nova funcionalidade a ser integrada na versão final. Os pares serão constituídos por um programador encarregue de escrever o código e por alguém mais concentrado na visão geral da aplicação. Tipicamente, os pares de programação concentrar-se-ão em pequenas unidades funcionais e farão o seu desenvolvimento utilizando o *Test-Driven Development*. Estas características permitem planear pequenas *releases* muito rapidamente que podem ser mostradas ao cliente, mantendo a sua confiança no desenvolvimento do software. O *Test-Driven* Software permite também implementar funcionalidades menos propensas a bugs.

Uma outra característica inerente ao XP, prende-se com conhecimento de todo o código por parte de todos os programadores. O *Collective Code Ownership*, como é chamado, é obtido com a rotação dos papéis nas sessões de *Pair Programming*. Assim, todos conhecem o código e todos são capazes de o modificar quando necessário, caso exista algum erro por exemplo. Esta característica permite obter maior flexibilidade na implementação do produto final, acelerando o processo.

Nesta metodologia, a programação de novas funcionalidades é feita sempre sob a versão mais recente da aplicação. Assim, todos os programadores devem, assim que terminam o desenvolvimento de determinada parte da aplicação, fazer o upload da nova versão para o repositório de modo que as novas funcionalidades estejam disponíveis para outras equipas de desenvolvimento. Isto permite evitar problemas de integração no final do projecto.

3.2. Testes a Efectuar

Ao longo de todo o processo de desenvolvimento de software, deverão ser efectuados os seguintes testes de forma a assegurar a qualidade do produto e a correcta integração de todos os componentes desenvolvidos.

3.2.1. Testes Unitários

Serão efectuados testes unitários sobre todos os componentes resultantes do processo de *Pair Programming*, assegurando que todas as unidades integrantes (métodos/funções) se encontram livres de erros. De facto, os testes unitários serão efectuados antes de própria implementação, seguindo assim o *Test-Driven Development*.

Estes testes permitirão, por um lado, revelar erros em funções e, por outro, revelar erros no funcionamento do componente como um todo. Serão efectuados pelo próprio programador no processo de implementação.

3.2.2. Testes de Integração

Após a integração de uma unidade funcional, serão efectuados testes que permitam revelar erros na integração do novo componente com o resto da aplicação. Assegura-se deste modo a correcta interligação entre todos os componentes existentes.

3.2.3. Testes de Usabilidade

De modo a assegurar o desenvolvimento de uma interface de utilizador altamente usável e identificar erros no fluxo de informação da interface, serão aplicados testes de usabilidade sobre o produto de software.

3.2.4. Testes de Aceitação

Conjuntamente com o cliente, serão construídos e aplicados testes de aceitação sobre o produto de software que assegurem a conformidade da aplicação com os requisitos impostos pelo cliente.

3.3. Gestão de Riscos

A gestão de riscos, sua identificação e mitigação, é um processo fundamental no processo de desenvolvimento do projecto. O sucesso do projecto depende em grande parte da capacidade da equipa em identificar riscos e elaborar planos a *priori* capazes de os eliminar caso estes se verifiquem. Para que a gestão de riscos seja eficaz, será fundamental e indispensável a disponibilidade de toda a equipa, assim como a dos nossos clientes e supervisor, para que existam flexibilidade e capacidade de resposta imediata a situações de risco.

3.3.1. Políticas de Gestão de Riscos

Para assegurar a correcta gestão dos riscos, a µTech segue um conjunto de princípios fundamentais que lhe permitem elaborar um plano eficaz para lidar com riscos no decorrer de todo o processo de desenvolvimento.

Análise Inicial dos Riscos do Projecto

Antes do arranque do projecto, é efectuado um levantamento dos riscos que poderão surgir no decorrer de todo o projecto. Esta análise inicial permite à equipa melhor tomar consciência do grau de complexidade do projecto e prepará-la já aí para possíveis dificuldades.

Revisão e Controlo Frequentes

Durante o planeamento das tarefas semanais, é efectuado um levantamento dos riscos que realmente se verificaram e, novamente, é feito uma análise dos riscos passíveis de se verificarem no decorrer dos trabalhos da próxima iteração.

Reacção e Comunicação Imediatas

Quando existe uma situação de risco que se concretiza, esta deve ser imediatamente reportada ao resto da equipa. Aqui é importante minimizar o tempo de resposta à situação verificada, sendo para isso importante a disponibilidade de todos os elementos. Quanto mais depressa a situação for reportada, mais depressa as acções necessárias para mitigar a situação podem ser tomadas de modo a que o processo de desenvolvimento planeado seja afectado o mínimo possível.

Participação Conjunta

Todos os *stakeholders* do projecto devem participar nas acções relativas aos pontos acima descritos.

3.3.2. Identificação e avaliação dos riscos

A identificação e avaliação de riscos são feitas com base num conjunto de parâmetros bem definidos e normalizados. Estes permitem obter um processo sistemático de processamento de riscos favorecendo a correcta identificação dos mesmos assim como uma boa forma de documentação. Assim, o processamento de riscos passa pela identificação dos seguintes parâmetros:

Tipo

Problema ou Risco. Indica se o que se vai definir a seguir se trata de um risco para o projecto, com potencial para se concretizar, ou se trata de um problema identificado e com necessidade de ser resolvido.

Data de Criação

Data de identificação do Risco ou Problema

Probabilidade

Probabilidade de o Risco se concretizar:

- Alta
- Média
- Baixa

Impacto

Impacto que o Risco terá no desenvolvimento do produto caso se concretize:

- Alto
- Médio
- Baixo

Descrição

Descrição do Risco ou Problema, descrevendo a situação em que se poderá verificar, em que parte do desenvolvimento e sob que condições.

Responsável

Responsável pelas acções a tomar para evitar a situação ou pela mitigação do problema caso este se concretize.

Acção/Mitigação

Acção específica a tomar para evitar a situação de risco e/ou acções que previnam a situação.

Data de Resolução

Data de resolução do Problema.

3.3.3. Responsabilidades

Para que exista uma correcta gestão dos riscos, a boa organização da equipa é indispensável, requerendo a atribuição de tarefas-chave a alguns elementos no processo de gestão e identificação de riscos.

Gestor do Projecto

Como gestor, cabe-lhe o papel da gestão global da gestão dos riscos:

- Delinear a estratégia base de gestão de riscos;
- Atribuir responsabilidades aos diversos membros para a implementação de um plano eficaz de identificação e mitigação de riscos;
- Responsabilidade central na comunicação de situações de risco a todos os *stakeholders* do projecto, para que seja possível a tomada de acções rápidas.

Gestor da Qualidade

Será o principal responsável pela elaboração de um bom plano de gestão de riscos, capaz de mitigar situações que afectem potencialmente qualquer artefacto produzido ao longo do projecto:

- Gerir o documento que contém todos os riscos identificados ao longo do projecto;
- Avaliação do processo de gestão de riscos;
- Revisão periódica para manutenção dos parâmetros de qualidade dos processos do projecto e dos riscos identificados.

Equipa

Como um todo, será responsável pela identificação de riscos potenciais assim como de riscos concretizados e da implementação de planos de contenção de riscos:

- Comunicação imediata de novos riscos identificados ao resto da equipa;
- Colocação em prática de processos de mitigação de riscos;
- Avaliação periódica dos riscos.

Clientes

Como principais interessados no sucesso do projecto, terá também um papel preponderante no processo de gestão de riscos:

- Ajudar na identificação de riscos;

- Dar prioridade a resolução de situações de risco;
- Comunicação constante com a equipa de modo a ajudar a definir o processo de gestão de riscos.

3.4. Fases do Projecto

O projecto "Módulo de Configuração do *CubeNavigator*" é constituído por duas fases distintas dentro da µTech. De seguida, evidencia-se cada fase, acções a serem tomadas e artefactos a serem produzidos:

Fase de Concepção

Deve ser efectuado:

- Realização da reunião de arranque do projecto;
- Planeamento inicial do projecto;
- Planeamento semanal do projecto;
- Definição de actividades e estimação;
- Execução do plano;
- Monitorização e controlo do progresso e de custos;
- Garantia e controlo da qualidade;
- Monitorização do risco;
- Reporte semanal do estado do projecto;

Deve ser produzido:

- Plano Inicial do Projecto;
- Relatório de Especificação de Requisitos;
- Plano de Testes de Aceitação;
- Protótipo (completo a nível de interface);
- Relatório Preliminar de Definição da Arquitectura;
- Manual de Qualidade;
- Relatório Intermédio de Execução de Projecto;
- Plano de Projecto para a segunda fase do projecto;

Fase de Desenvolvimento

Deve ser efectuado:

- Planeamento semanal do projecto;
- Definição de actividades e estimação;

- Execução do plano;
- Monitorização e controlo do progresso e dos custos;
- Garantia e controlo da qualidade;
- Monitorização do risco;

Deve ser produzido:

- Pacotes de Instalação e Manutenção do Produto;
- Manual do Utilizador;
- Versões Finais de Relatório de Especificação de Requisitos, Relatório de Definição da Arquitectura e Manual de Qualidade;
- Relatório de Desenvolvimento;
- Relatório de Testes;

4. Definição de processos

As *deliverables* a serem desenvolvidas durante o projecto devem respeitar certos requisitos descritos nas *process scripts* retratadas neste tópico do manual da qualidade. Os *stakeholders* envolvidos na criação do respectivo *deliverable* devem ter a *process script* como um guia, antes de o seu desenvolvimento certificar que os critérios de entrada são cumpridos e seguir todos os passos/actividades que lhe está atribuído.

4.1. Objectivos

Este tópico do manual foca na definição de *process scripts* para as *deliverables* mais importantes do projecto. A definição de um processo para desenvolver uma *deliverable* garante a sua qualidade através de um conjunto de actividades aplicadas sistematicamente pelos respectivos intervenientes respeitando determinados critérios de entrada e saída.

Cada *process script*, representado por uma tabela, mantém informação relativa ao seu objectivo, actividades, *stakeholders* ou intervenientes, critérios de entrada e de saída.

4.2. Plano do Projecto

Process Script	
Objectivo	Desenvolver um plano que permite gerir e controlar as diferentes fases do projecto
Critérios de entrada	-Descrição da aplicação a ser desenvolvida -Cargos da empresa definidos -Manual de qualidade
Passos	**Actividades**
1.	Obter um acordo na especificação de requisitos
2.	Estabelecer estimativas das tarefas e tamanho das *deliverables*
3.	Preencher um formulário ou *Workbook* com dados do plano
4.	Rever o plano do projecto com o cliente e obter um acordo
5.	Registar o tempo gasto ao longo do projecto
Critérios de saída	Plano do projecto completo com dados estimados e actualizados
Stakeholders	-Gestor do Projecto -Gestor do Planeamento -Cliente

Tabela 2 - *Process Script* Plano do projecto

4.3. Relatório de especificação de requisitos

Process Script	
Objectivo	Documentar a descrição das funcionalidades, dos requisitos funcionais e não funcionais da aplicação a ser desenvolvida
Critérios de entrada	-Descrição da aplicação a ser desenvolvida -Manual de qualidade
Passos	**Actividades**
1.	Identificar funcionalidades da aplicação a ser desenvolvida
2.	Identificar e analisar os requisitos funcionais recorrendo a modelo de casos de utilização
3.	Identificar e analisar os requisitos não funcionais (propriedades e restrições da aplicação)
4.	Identificar e analisar os requisitos de informação
5.	Validar os requisitos especificados com o cliente
Critérios de saída	Relatório de especificação de requisitos
Stakeholders	-Gestor da Interacção com Cliente e Requisitos -Cliente

Tabela 3 - *Process Script* Relatório de especificação de requisitos

4.4. Plano de testes de aceitação

Process Script	
Objectivo	-Definir um conjunto de casos de teste. -Identificar procedimentos de teste que especificam como devem ser realizados os casos de teste.
Critérios de entrada	-Relatório de Arquitectura -Relatório de especificação de requisitos -Manual de qualidade
Passos	**Actividades**
1.	Analisar requisitos e funcionalidades da aplicação
2.	Identificar e descrever os casos de teste
3.	Identificar e estruturar procedimentos de teste
4.	Rever os testes
Critérios de saída	Plano de testes de aceitação
Stakeholders	-Gestor dos testes -Cliente

Tabela 4 - *Process Script* Plano testes de aceitação

4.5. Protótipo de interface

Process Script	
Objectivo	Desenvolver um protótipo de interface funcional, fácil de utilizar e que vai de encontro aos requisitos especificados
Critérios de entrada	-Relatório de especificação de requisitos -Relatório de arquitectura -Manual de qualidade
Passos	**Actividades**
1.	Projectar o protótipo de interface
2.	Implementar o protótipo de interface
3.	Obter o parecer do cliente sobre o protótipo de interface desenvolvido
Critérios de saída	Obter um protótipo de interface com boa usabilidade
Stakeholders	-Gestor de Usabilidade e Desenho de Interfaces -Cliente

Tabela 5 - *Process Script* Protótipo de interface

4.6. Relatório de arquitectura

Process Script	
Objectivos	-Definir uma arquitectura inicial com base em sistemas semelhantes. -Determinar os padrões de arquitectura -Definir a estratégia de reutilização -Disponibilizar dados para o processo de planeamento
Critérios de entrada	-Relatório de especificação de requisitos -Manual de Qualidade -*Software Architecture* RUP -Modelo 4+1 de Krutchen
Passos	**Actividades**
1.	Desenvolver visão geral da arquitectura
2.	Especificar os requisitos da arquitectura
3.	Criar uma estrutura inicial do modelo de arquitectura
4.	Desenvolver o modelo de implementação
5.	Identificar mecanismos de análise que ofereçam suporte a recursos contidos nos requisitos funcionais básicos do sistema
6.	Criar realizações de casos de utilização
7.	Identificar Interacções Estereotipadas capturadas como realizações de casos de utilização
8.	Rever os resultados da análise da arquitectura
Critérios de saída	Relatório de arquitectura
Stakeholders	Gestor de desenho

Tabela 6 - *Process Script* Relatório de arquitectura

4.7. Manual de qualidade

Process Script	
Objectivo	Desenvolver um documento especificando um conjunto de actividades que garantem a qualidade do produto desenvolvido
Critérios de entrada	-Cargos da empresa definidos -*Rational Unified Process* (RUP) -Melhores práticas TSP/PSP -Descrição da aplicação a ser desenvolvida -Modelo CMMI nível 2
Passos	**Actividades**
1.	Verificar a existência de objectivos de qualidade definidos para o projecto
2.	Definir papéis ou responsáveis para garantia da qualidade
3.	Especificar o processo de desenvolvimento de software
4.	Definir processos de desenvolvimento das *deliverables*
5.	Especificar standards de codificação, templates de documentação e métricas
6.	Definir *checklists* de revisão
7.	Especificar ferramentas ou tecnologias utilizadas no projecto
Critérios de saída	Manual de qualidade
Stakeholders	-Gestor da qualidade -Gestor do projecto -Cliente

Tabela 7 - *Process Script* Manual de qualidade

4.8. Relatório de execução da 1ª Fase do Projecto

Process Script	
Objectivo	Desenvolver um documento que apresente os resultados da fase de concepção do projecto
Critérios de entrada	Incluir todas as *deliverables* a serem entregues no fim desta fase
Passos	**Actividades**
1.	Descrever fases do projecto, *deliverables* e *stakeholders* envolvidos
2.	Descrever o modelo de gestão
3.	Descrever o plano inicial do projecto
4.	Analisar o trabalho realizado e avaliar o esforço da equipa
5.	Preparar o início da fase de desenvolvimento
Critérios de saída	Relatório de execução da 1ª fase do projecto
Stakeholders	Gestor do projecto

Tabela 8 - *Process Script* Relatório de execução da 1ª Fase do Projecto

4.9. Manual do utilizador

Process Script	
Objectivo	Disponibilizar ao utilizador um documento com a descrição da aplicação, para que serve, e como utiliza-la de forma eficiente
Critérios de entrada	-Relatório de especificação de requisitos -Relatório de arquitectura -Produto final -Manual de Qualidade
Passos	**Actividades**
1.	Especificar os requisitos do sistema
2.	Descrição das funcionalidades do sistema
3.	Desenvolver tutoriais para cada funcionalidade da aplicação
4.	Desenvolver soluções para problemas ou bugs que possam ser encontrados
Critérios de saída	Manual do utilizador
Stakeholders	-Gestor de suporte -Cliente

Tabela 9 - *Process Script* Manual do utilizador

4.10. Pacotes de Instalação e Manutenção do Produto

Process Script	
Objectivo	Disponibilizar um ficheiro com o nome da aplicação e versão, contendo os ficheiros necessários para instalar o produto e com o respectivo código fonte
Critérios de entrada	-Relatório de especificação de requisitos -Relatório de arquitectura -Manual de Qualidade
Passos	**Actividades**
1.	Criar um ficheiro de instalação da aplicação
2.	Criar um documento com instruções de instalação e configuração da aplicação
3.	Disponibilizar ficheiro com o código fonte e bibliotecas necessárias ao desenvolvimento da aplicação
Critérios de saída	Pacotes de Instalação e Manutenção do Produto
Stakeholders	Gestor de suporte

Tabela 10 - *Process Script* Pacotes de Instalação e Manutenção do Produto

4.11. Relatório de Desenvolvimento

Process Script	
Objectivo	Documentar o estado do desenvolvimento do projecto quanto da implementação dos casos de utilização e arquitectura especificados na fase de concepção
Critérios de entrada	-Relatório de especificação de requisitos -Relatório de arquitectura -Manual de Qualidade
Passos	**Actividades**
1.	Implementar a arquitectura
2.	Implementar os casos de utilização
3.	Testar o programa, corrigir e registar todos os defeitos encontrados
4.	Analisar e avaliar a fase de desenvolvimento
Critérios de saída	Relatório de Desenvolvimento
Stakeholders	Gestor de Implementação

Tabela 11 - *Process Script* Relatório de Desenvolvimento

4.12. Relatório de Testes

Process Script	
Objectivo	Documentar os resultados dos testes planeados e especificar os procedimentos de teste
Critérios de entrada	-Plano de testes de aceitação -Manual de Qualidade
Passos	**Actividades**
1.	Descrever resultados dos testes de componentes
2.	Descrever resultados dos testes usabilidade
3.	Descrever resultados dos testes de aceitação
Critérios de saída	Relatório de Testes
Stakeholders	Gestor dos testes

Tabela 12 - *Process Script* Relatório de Testes

5. Normas, Métricas e Análise

5.1. Objectivos

Este capítulo apresenta todas as métricas e normas associadas necessárias para quantificar a qualidade do produto obtido, dos processos utilizados e ainda do desempenho da equipa. Estas métricas deverão ser tidas em conta no decorrer de todo o projecto.

5.2. Normas de Documentação

Todos os documentos produzidos no decorrer do projecto devem seguir os templates do manual de imagem da µTech. Além disto, devem também seguir a seguinte estrutura:

- Capa
  - Título do Documento
  - Nome do projecto
  - Responsável
  - Data
- Histórico de Alterações ao Documento
- Resumo
- Conteúdo
  - Índice
  - Índice de figuras
  - Índice de tabelas
- Corpo do Documento
- Glossário
- Referências

5.3. Normas de Codificação

De forma a garantir a qualidade do código gerado durante todo o processo de implementação do produto, a µTech seguirá algumas normas de codificação. Estas baseiam-se em grande parte nas convenções de escrita de código da framework *Java* [1].

Nomes de Ficheiros

Os nomes dos ficheiros Java criados devem ter o mesmo nome que a classe pública implementada no ficheiro.

Nomes de Packages

Os nomes das packages devem ser escritos totalmente em minúsculas, devendo o prefixo de todas as packages ser "*pt.miutech*", seguido do nome do produto. A organização mais interior deve obedecer à própria arquitectura do produto, sendo que qualquer package deve continuar a ser escrita em minúsculas.

Comentários

Todos os comentários deverão seguir os standards de javadoc. Devem ser tidos em conta os comentários de documentação, delimitados por **/**...*/** e os comentários de implementação, delimitados por **/*...*/**.

Assim será possível gerar documentação para todo o código implementado e manter também alguns comentários mais específicos acerca da implementação e mais direccionados para programadores.

Comentários JavaDoc

Antes da implementação de uma classe, deve existir um comentário descritivo da classe e da sua função. Devem também ser indicadas as versões e os nomes do autor através das tags *@author* e *@version*.

Todas as variáveis estáticas, publicas e privadas de uma classe devem ter um comentário antes da sua declaração descrevendo o objectivo da variável.

Todas as funções públicas e privadas devem também ter um comentário imediatamente antes da sua declaração descrevendo sumariamente o seu objectivo assim como os seus parâmetros de entrada e o seu valor de retorno. Estes podem ser descritos através de tags específicas: @param e @return respectivamente.

Outro tipo de varáveis podem ter comentários descritivos sob a forma de comentários de implementação quando for necessário, mas não devem ser usados comentários javadoc.

Indentação

Todo o código deve ser indentado com base em 4 espaços.

Declarações

Devem-se evitar declarações múltiplas numa única linha, uma vez que não favorecem a escrita de comentários.

As declarações devem, preferencialmente, existir no início de um bloco como, por exemplo, um bloco *if* ou função.

Os nomes de classes ou interfaces devem começar todas com uma letra maiúscula e sempre que haja necessidade de fazer algum tipo de separação no nome por questões de legibilidade devem ser utilizadas letras maiúsculas para fazer a separação, ao invés de *underscores*. Por exemplo: *MyClassName* e não *My_class_name*. Os nomes das classes devem também corresponder a nomes no contexto gramatical e não verbos.

Os nomes de métodos devem ser verbos. Devem ser iniciados por uma palavra em minúsculas e se for necessário criar um nome composto, então usar uma letra maiúscula para fazer a separação: *criar* ou *criarGrafo* ou *criarGrafoDirigido* e não *Criar* ou *Criargrafodirigido* ou *Criar_grafo_dirigido*.

Os nomes de variáveis internas devem ser curtos e com significado imediato. Devem seguir a mesma regra das funções em relação à composição de nomes. Devem ser evitados nomes demasiado curtos, como *i* ou *j*, tipicamente usados para variáveis temporárias dentro de blocos.

As constantes, de qualquer tipo, devem ser escritas em maiúsculas e, quando existe necessidade de criar um nome composto, então devem ser utilizados *underscores* para fazer a separação de nomes: *MIN_WIDTH*, *MAX_WIDTH*.

Os blocos que necessitam de chavetas, devem utilizar a primeira chaveta à direita da declaração do bloco, separado por um espaço e não na linha abaixo da declaração:

```
1   /**
2    * OK!
3    */
4   void block() {
5   }
```

```
/**
2    * NOT OK!
```

```
3    */
4   void block()
5   {
6   }
```

Tamanhos

Devem-se evitar ficheiros e funções com demasiadas linhas de código. Assim, a µTech aconselha aos seus programadores a escrita de ficheiros com não mais de 500 linhas de código e funções com tamanho entre 30 e 50 linhas, de modo a favorecer a legibilidade. Assim, é aconselhado que funções mais complexas e com maior número de linhas de código sejam distribuídas por funções auxiliares.

Código gerado automáticamente, por exemplo, através de ferramentas para design de interfaces gráficas, é excluído das regras acima citadas para os tamanhos das funções e ficheiros.

5.4. Normas de controlo de versões e alterações

O processo de controlo de versões e alterações a ser seguido neste projecto baseia-se nas principais actividades da área de processo gestão de configurações (modelo CMMI nível 2).
A gestão de configurações permite estabelecer e manter a integridade de produtos desenvolvidos no projecto utilizando identificação, controlo, estado, e auditoria das diferentes configurações e alterações.

Principais actividades no controlo de versões e alterações:

- Identificar itens de configuração
- Estabelecer um sistema de gestão de configuração
- Estabelecer *baselines* (pontos de referência de uma configuração)
- Controlar mudanças e itens de configuração
- Registar o estado da configuração
- Realizar auditorias de configuração

A ferramenta a ser utilizada neste projecto para este propósito é o SVN.
O SVN (*subversion*) é um sistema de controlo de versões que mantém um historial de todo o trabalho e alterações efectuadas pelos membros do grupo, para um determinado conjunto de ficheiros.

5.5. Métricas

As métricas são necessárias como forma de adquirir dados que possam vir a ser utilizados no futuro para melhorar a qualidade dos processos da empresa.

Assim, serão adquiridas métricas nas seguintes áreas:

- Qualidade do Produto
- Qualidade dos Processos
- Qualidade dos Testes

Qualidade do Produto

A qualidade do produto, sendo algo sempre muito subjectivo, será mensurada através de dois componentes: a satisfação do cliente e a densidade de defeitos.

A densidade de defeitos será o parâmetro mais objectivo, e pode ser calculado através do número de linhas de código que compõem o produto e o número de defeitos encontrados: **DD = ND / LOC**

- DD: densidade de defeitos;
- ND: número de defeitos;
- LOC: número de linhas de código (*lines of code*);

A satisfação do cliente pode ser parametrizada através de inquéritos efectuados após a utilização do produto. Os inquéritos deverão ser dar a hipótese de resposta com base numa escala de satisfação (1 – 4):

- Muito Satisfeito (4)
- Satisfeito (3)
- Pouco Satisfeito (2)
- Insatisfeito (1)

Estes questionários devem questionar o cliente acerca de vários aspectos do produto e apreciação global será a média dos resultados nos vários aspectos.

Qualidade dos Processos

Aqui é necessário verificar a relação existente entre os requisitos identificados e aqueles que foram realmente implementados. Para isso utiliza-se a seguinte relação:

RR = RA/RI * 100

- RR: Rastreamento de Requisitos
- RA: total de requisitos implementados na solução final;
- RI: total de requisitos identificados;

Esta fórmula resulta num valor percentual que, no caso em que a aplicação contempla requisitos adicionais, não identificados inicialmente, pode ser maior que 100%.

Qualidade dos Testes

Esta área é composta por várias métricas individuais. Primeiro é necessário identificar e documentar todos os defeitos encontrados. Por outro lado, é também necessário catalogar os defeitos encontrados segundo uma tabela de gravidade. A tabela utilizada é a que a seguir se apresenta:

Gravidade	Descrição
Nível 1	Defeito crítico. Impede a utilização correcta da aplicação. Provoca anomalias graves.
Nível 2	Defeito grave. Não funciona como requerido. Não impede a utilização mas tem impacto nos processos.
Nível 3	Defeito leve. Não afecta a utilização normal, mas a falha tem de ser resolvida
Nível 4	Melhoria. Novos pedidos face ao âmbito do projecto. Melhorias cosméticas, de interface.

Tabela 13 - Defeitos e Gravidades

Por fim, é necessário encontrar uma medida de correcção de defeitos. Para isso, calcula-se a percentagem de defeitos corrigidos:

- TCD = (DC / DE) * 100
- TCD: Taxa de correcção de defeitos;
- DCT: Defeitos corrigidos;
- DE: Defeitos encontrados.

5.6. *Checklists* de Revisão de Código

A seguir se apresenta listagens de itens a serem tidos em conta para rever o código gerado. Estes itens

são guias que asseguram a algum nível a qualidade do código desenvolvido.

Estrutura

1	O código implementa totalmente e correctamente a arquitectura proposta?
2	A escrita do código segue os standards acordados?
3	O código encontra-se bem estruturado, consistente no estilo e bem formatado?
4	Existe alguma função ou bloco de código que nunca é executado?
5	Existe alguma função ou bloco de código escrito com o intuito de efectuar algum teste que não foi removido?
6	Existe alguma função ou bloco de código que possa ser substituído por uma chamada a uma biblioteca ou componente reutilizável?
7	Existem blocos de código repetido que possam ser condensados numa única função?
8	A utilização da memória é eficiente?
9	Existe algum componente demasiado complexo e que possa ser dividido em múltiplos componentes ou possa ser reestruturado?

Tabela 14 - *Checklist* de revisão da estrutura do código

Documentação

1	Os standards de documentação e comentários estão a ser correctamente utilizados?
2	Os comentários estão de acordo com o código?

Tabela 15 - *Checklist* de revisão da documentação do código

Variáveis

1	As variáveis estão devidamente definidas com nomes com significado apropriado, consistentes e claros?
2	Todas as variáveis atribuídas têm tipo consistente?
3	Existem variáveis redundantes ou nunca utilizadas

Tabela 16 - *Checklist* de revisão das variáveis do código

Operações Aritméticas

1	O código evita fazer comparações de igualdade com número de vírgula flutuante?
2	O código evita erros de arredondamento?
3	Existem divisões por zero?
4	É usada aritmética inteira ou de ponto flutuante de acordo com o resultado pretendido?
5	São usados os operadores de comparação correctos para o propósito a que se destinam?

Tabela 17 - *Checklist* de revisão de operações aritméticas do código

Ciclos

1	Todos os blocos lógicos e de ciclos se encontram devidamente encadeados?
2	Todos os casos base se encontram testados em primeiro lugar em estruturas *if* ou *else-if*?
3	Todos os casos são testados nos blocos *if*, *else-if* ou *switch*?
4	Os *switch* têm sempre um *default*?
5	Todos os índices são devidamente inicializados imediatamente antes do ciclo?
6	Existe alguma declaração dentro de um *loop* que possa ser colocada fora do *loop*?
7	O código de *loop* evita manipular a variável índice ou usá-la fora do *loop*?
8	Existe a possibilidade de ciclos infinitos?

Tabela 18 - *Checklist* de revisão de ciclos do código

Programação Defensiva

1	Todos os índices utilizados em acesso a posições de memória são testados em relação ao limite do *array* ou ficheiro?
2	Os argumentos de entrada são validados?
3	As variáveis de retorno são atribuídas?
4	A informação correcta é utilizada em cada instrução?
5	Os ficheiros são testados para verificar a sua existência antes da tentativa de acesso?
6	Todos os ficheiros são deixados num estado consistente após a terminação do programa?
7	Existe algum método que altere valores que apenas devia usar para leitura?
	Estão a ser tratados devidamente os erros de *input/output*?
9	Estão a ser apresentadas mensagens de erro e de aviso?

Tabela 19 - Checklist de revisão de programação defensiva do código

6. Ferramentas

Aqui se apresentarão as principais ferramentas utilizadas pela empresa durante todo o projecto, assim como as características que justificam a sua utilização.

6.1. Ferramentas Utilizadas

Redmine

Uma ferramenta *web* para gestão de projectos, extremamente flexível. Permite-nos gerir todas as tarefas da equipa assim como inserir *milestones*, correspondentes aos *deliverables*.

Sendo a principal função da ferramenta a gestão de projectos, com integração com controlo de versões e gestão de tarefas, incorpora ainda muitas outras funcionalidades com grande utilidade para a equipa:

- Inserção de documentos, onde a equipa pode inserir periodicamente os planos semanais, actas ou outros documentos;
- *Wiki* integrada, onde a equipa pode criar toda a documentação necessária colaborativamente e com maior facilidade;
- Calendário para gestão de tarefas inseridas;
- Criação automática de diagramas de Gantt, a partir das tarefas criadas.

Mailing-List* do *Google Groups

Esta *mailing-list* dá-nos grande flexibilidade na comunicação entre os elementos da equipa e também quando é necessário estabelecer contactos exteriores à equipa: toda a informação é automaticamente difundida para todos os elementos. Permite-nos ainda alojar alguns documentos resultantes do trabalho periódico da equipa.

Blog

O *blog* é uma ferramenta útil para difundir informação para o exterior. Os principais desenvolvimentos relativos ao estado do projecto são lá colocados, estando visíveis para todos os interessados na equipa. É uma forma de publicitar o trabalho efectuado e manter os clientes e principais *stakeholders* do projecto informados.

6.2. Linguagens Utilizadas

A aplicação a desenvolver será totalmente criada em Java. Após a consideração das hipóteses disponíveis, e também por sugestão do cliente, decidiu-se pela linguagem Java, que nos possibilita a construção de uma aplicação com uma interface gráfica bastante apelativa e dinâmica sem necessidade de recorrer a qualquer tipo de bibliotecas ou ferramentas externas. Para além disto, a necessidade de ligação e extracção informação de uma base de dados é-nos facilitada através da utilização de Java: JDBC, parte integrante da biblioteca Java, permite-nos a ligação a base de dados de forma relativamente fácil e transparente. Uma outra justificação para a utilização desta linguagem advém ainda da necessidade da criação e processamento de XML: a extensa biblioteca de Java contém algumas classes que facilitam o processamento de XML.

6.3. Ferramentas de Desenvolvimento

A ferramenta de desenvolvimento de código será o *Netbeans IDE 6.5*. A principal justificação para a escolha deste IDE advém do seu poderoso editor gráfico para aplicações *Java-Swing*.

7. Glossário

- **Blog** – Abreviação de *weblog*, é uma página *Web* actualizada regularmente por uma pessoa ou um grupo em que é possível publicar qualquer tipo de conteúdo, de textos a vídeos. Com ferramentas próprias de iteração, por meio de comentários adicionados por qualquer leitor a cada novo *post* ou actualização.
- **Collective Code Ownership** – Prática do *Extreme Programming* em que todos são responsáveis pelo código desenvolvido, ou seja, todos podem alterar qualquer parte do código, o que permite uma evolução rápida do sistema, em caso de erros no código qualquer programador pode corrigi-lo.
- **CMMI** – *Capability Maturity Model Integration* é um modelo de referência que contém práticas (Genéricas ou Específicas) necessárias à maturidade em disciplinas específicas.
- **Deliverable** – Artefacto do projecto resultante de uma tarefa ou conjunto de tarefas que deve ser entregue numa data especifica.
- **Métrica** – Medida de alguma propriedade ou especificação do software, permite calcular orçamentos, avaliar o desempenho dos programadores, controlar número de erros ou bugs.
- **Milestone** – Designação de um ponto de controlo em um cronograma, representa a conclusão de um conjunto de tarefas ou fase, passiva de aprovação e formalização por parte do cliente.
- **Norma** – Conjunto de procedimentos, elaborado por um grupo ou instituto certificado, que estabelecem um modelo de gestão da qualidade para as organizações ou empresas na geração de documentos e desenvolvimento da sua actividade.
- **Release** – Designação dada a criação e disponibilização de uma nova versão de um produto de software.
- **Stakeholders** – Termo utilizado em administração que refere-se as pessoas, instituições ou empresas envolvidas em acções de uma determinada organização.
- **Test-Driven Development** – Técnica de desenvolvimento de software que utiliza pequenas iterações de desenvolvimento baseadas na pré-definição de casos de teste que definem melhorias pretendidas e novas funções.
- **Workbook** – Folhas de cálculo Excel com dados relativos ao plano do projecto, permite especificar estimativas, planear tarefas, fazer *tracking* do projecto.
- **Wiki** – Sítio *Web* que permite alterar, apagar, reescrever ou adicionar conteúdos sem a necessidade de muitos conhecimentos técnicos, é muito utilizado para a criação interactiva de documentos, reforça a ideia de colaboração e troca de conhecimento, é uma forma fácil de compartilhar informações.

8. Referências

1. http://java.sun.com/docs/codeconv/: Code Conventions for the Java Programming Language
2. http://paginas.fe.up.pt/~jpf/teach/LES/template_arquitectura.html: Faria, J. P., & Aguiar, A. (s.d.). Estrutura típica da documentação da fase de desenho de alto nível.
3. http://www.wthreex.com/rup/: Rational Unified Process
4. Sommerville, I. (2004). Software Engineering.